Introduction à la cybersécurité - Lab1

Remarques importantes :

- Tout d'abord, nous vous donnerons la syntaxe exacte des commandes afin de vous donner une idée générale du fonctionnement des commandes OpenSSL.
commandes OpenSSL fonctionnent. Ensuite, vous devrez faire des recherches sur Internet pour trouver la bonne syntaxe et répondre aux questions.
-Ce laboratoire doit être réalisé par une équipe de deux étudiants.
-Vous devez fournir un rapport unique sur votre travail, intitulé avec vos noms et prénoms.
- Le rapport doit contenir des captures d'écran de toutes les parties avec *.
- N'oubliez pas d'indiquer votre nom et votre numéro de groupe dans le rapport.
- Le rapport doit être téléchargé sur Moodle avant la fin de la session.

1 Introduction à OpenSSL

1. OpenSSL est une boîte à outils cryptographique mettant en œuvre le protocole SSL/TLS. Il offre une bibliothèque de programmation en C pour construire des applications client/serveur sécurisées basées sur le protocole SSL/TLS. Une commande en ligne OpenSSL permet la création de clés, la création d'un certificat numérique, le calcul d'un hash, le cryptage et le chiffrement asymétrique, etc. Vous trouverez toutes les informations sur OpenSSL à l'adresse suivante https://www.openssl.org/

2. La syntaxe générale d'une commande openssl est : **$ openssl command option**

3. Les commandes OpenSSL à utiliser dans ce laboratoire sont :
- **genrsa** : permet de générer une paire de clés (clé privée et clé publique)
- **rsautl** : permet d'effectuer un cryptage et un décryptage asymétrique. Elle permet également de signer des données et de les vérifier.
- **dgst** : permet de générer un hash.
- **rsa** : permet d'extraire la clé publique d'un fichier contenant une paire de clés.

4. Une commande OpenSSL possède plusieurs options. Les options de chaque commande sont très importantes à consulter. Pour cette raison, afin de consulter les options de la commande :
- **genrsa**, veuillez utiliser la commande : **openssl genrsa -help**
- **rsautl**, veuillez utiliser la commande : **openssl rsautl -help**

- **dgst**, veuillez utiliser la commande : **openssl dgst -help**
- r**sa**, veuillez utiliser la commande : **openssl rsa -help**

5. **Veuillez suivre les étapes suivantes pour commencer vos travaux pratiques** :
(a) Lancez le système d'exploitation Debian ou la machine virtuelle Debian.
(b) Ouvrez un **terminal**.
(c) Entrez la commande UNIX **sudo su** (mot de passe par défaut : root) ou demandez à votre superviseur de laboratoire de vous donner le bon mot de passe.
de vous donner le bon mot de passe.
(d) Créez un nouveau dossier nommé **LAB1** en entrant la commande UNIX : **mkdir LAB1**
(e) Accédez à ce dossier en entrant la commande UNIX : **cd LAB1**
(f) Dans la section 2, vous aurez besoin d'utiliser le chemin d'accès à un dossier. Par exemple, nous illustrons dans la Figure 1 le lien pour le dossier du LAB1 qui est : /home/**LAB1**. Afin de récupérer le chemin d'accès à votre dans le dossier **LAB1**, entrez la commande UNIX : **pwd**
(g) Créez un nouveau dossier nommé **Alice** en entrant la commande UNIX : **mkdir Alice**
(h) Créez un nouveau dossier nommé **Bob** en entrant la commande UNIX : **mkdir Bob**.
(i) Accédez au dossier d'**Alice** en entrant la commande UNIX : cd **Alice**
(j) Afin de récupérer le chemin d'accès à votre dossier **Alice**, entrez la commande UNIX : **pwd**
(k) Créez le fichier **AliceDocument** en entrant la commande UNIX : **gedit AliceDocument**
(l) Ecrivez un texte de votre choix, par exemple : **Bonjour Bob, je suis Alice**
(m) Sauvegardez les modifications et retournez au **Terminal.**

6. Assurez-vous que vous êtes dans le dossier d'**Alice**. Nous voulons créer une paire de clés pour Alice et pour cette raison nous allons utiliser spécialement la commande OpenSSL **genrsa** comme suit :
- Entrez la commande OpenSSL : **openssl genrsa -out AliceKeyPair**
- Le fichier **AliceKeyPair** contient une paire de clés pour **Alice** "une clé publique et une clé privée". Vérifiez que le fichier a été créé correctement en entrant la commande UNIX : **ls**

7. La clé publique d'**Alic**e est publique et peut donc être communiquée à n'importe qui. d'autre part. En revanche, le fichier **AliceKeyPair** contient également la clé privée, ce fichier ne peut donc être communiqué à personne et doit rester secret. Pour cette raison, nous allons utiliser une commande OpenSSL pour extraire la clé publique d'**Alice** et la stocker dans un autre fichier.

- Entrez la commande OpenSSL : **openssl rsa -in AliceKeyPair -pubout -out AlicePublicKey**
- Pour plus d'informations sur les options : **-in, -pubout, -out**, veuillez utiliser la commande : **openssl rsa -help**
- Le fichier **AlicePublicKey** contient la clé publique d'Alice. Vérifiez que le fichier a été créé correctement en entrant la commande UNIX : **ls**
- Maintenant le fichier **AliceKeyPair** contient seulement la clé privée d'Alice et nous pouvons le renommer
AlicePrivateKey en entrant la commande UNIX : mv **AliceKeyPair AlicePrivateKey**
- Vérifiez que le fichier a été renommé correctement en entrant la commande UNIX : **ls**
- Si vous souhaitez voir le contenu des fichiers **AlicePublicKey et AlicePrivateKey**, veuillez utiliser les commandes UNIX :
- **gedit AlicePublicKey**
- gedit AlicePrivateKey

2 Exercice "Cryptographie asymétrique" → (10 points)

Dans cet exercice, l'intérêt est de crypter un document avec une clé publique et de le décrypter grâce à la clé privée.

1. Nous voulons faire pour **Bob** les mêmes étapes que pour **Alice** :
- Vous devez retourner dans le dossier parent **LAB1** en entrant la commande UNIX : **cd**....
- Vous devez accéder au dossier de **Bob** en entrant la commande UNIX : **cd Bob**
- * Maintenant, nous vous demandons de répéter les mêmes étapes pour Bob que nous avons fait pour Alice. A la fin, vous devez avoir deux fichiers pour **Bob : BobPublicKey et BobPrivateKey** → (1,5 pt).

2. Alice veut chiffrer **Alice Document** grâce à la clé publique de Bob, pour cette raison vous devez procéder comme suit :

- Assurez-vous que vous êtes dans le dossier de **Bob**. Afin de récupérer le chemin d'accès à votre dossier Bob, entrez la commande UNIX commande UNIX (voir section 1) : **pwd**

- Alice ne possède pas la clé publique de Bob, vous devez alors copier la clé publique de Bob dans le dossier d'Alice en entrant la commande cp **/home/UserName/LAB1/Bob/BobPublicKey /home/UserName/LAB1/Alice/**

- * Retourner dans le dossier parent **LAB1** et accéder au dossier **d'Alice** → (0.25 pt).
- * Vérifiez que **BobPublicKey** a été correctement copié dans le dossier d'Alice → (0,25 pt).
- Vous allez maintenant procéder au chiffrement du document **AliceDocument** grâce à **BobPublicKey** en nommant le document chiffré **AliceDocumentEncrypted**. Pour ce faire, veuillez entrer la commande OpenSSL :

openssl rsautl -encrypt -in AliceDocument -pubin -inkey BobPublicKey -out AliceDocumentEncrypted

- Pour plus d'informations sur les options : **-encrypt, -in, -pubin, -inkey, -out,** veuillez utiliser la commande commande : **openssl rsautl -help**
- * Vérifier que AliceDocumentEncrypted a été créé correctement → (0.25 pt).
- * Vérifiez le contenu du fichier **AliceDocumentEncrypted**. Que constatez-vous ? → (0.25 pt).

3. L'objectif dans cette étape est de décrypter **AliceDocumentEncrypted grâce à BobPrivateKey :**

- * Assurez-vous que vous êtes dans le dossier d'Alice. Nous vous demandons de copier **AliceDocumentEncrypted** dans le dossier de Bob. → (0,5 pt).
- * Retournez dans le dossier parent **LAB1** et accédez au dossier de **Bob** → (0,25 pt).
- * Vérifiez que **AliceDocumentEncrypted** a été copié correctement dans le dossier de Bob → (0.25pt).
- * Vous allez maintenant procéder au déchiffrement **de AliceDocumentEncrypted** grâce à **BobPrivateKey** en nommant le document décrypté **AliceDocumentDecrypted.** Pour ce faire, veuillez utiliser la commande **OpenSSL** openssl rsautl et les options : **-decrypt, -in, -inkey, -out → (2 pt).**
- * Vérifiez que le fichier **AliceDocumentDecrypted** a été créé correctement → (0,25 pt).
- * Vérifiez le contenu du fichier **AliceDocumentDecrypted.** Que constatez-vous ? → (0,25pt).

4. L'objectif de cette étape est de vous montrer que le chiffrement asymétrique ne peut pas être appliqué à des fichiers volumineux :

- * Retournez dans le dossier parent **LAB1** et accédez au dossier **d'Alice** → (0,25 pt).
- Créez un fichier de grande taille nommé **LargeFile** en entrant la commande OpenSSL :

openssl rand **-out LargeFile -base64 $((2**30 * 3/4))**

- * Essayez maintenant de crypter **LargeFile en utilisant BobPublicKey et en donnant un nom au fichier crypté LargeFileEncrypted** → (0,5 pt).

- Vous remarquerez que le chiffrement asymétrique n'est pas possible sur les grands fichiers. Cela s'explique par le fait que le chiffrement asymétrique coûte cher en puissance de calcul et est donc fait pour chiffrer de petites quantités de données.
C'est pourquoi il est recommandé d'utiliser le cryptage symétrique pour les gros fichiers. Pour plus d'informations sur la comparaison entre les deux cryptographies, veuillez visiter : **Lien 1, Lien 2**

5. L'objectif de cette étape est de vous montrer comment **Alice** peut générer une signature électronique afin de : s'authentifier auprès de **Bob**, assurer la non-répudiation pour elle-même et garantir l'intégrité des données signées.
- Assurez-vous que vous êtes dans le dossier d'**Alice**. Créez un fichier nommé **AuthData** et écrivez un texte de votre choix → (0,25 pt).
- * Vérifiez que **AuthData** a été créé correctement → (0,25 pt).
- * Copiez **AlicePublicKey** dans le dossier de **Bob** → (0,25 pt).
- Vous allez maintenant procéder à l'application de la fonction de hachage **SHA256** sur le document **AuthData** pour trouver son hachage **HashAuthData.** Pour ce faire, veuillez entrer la commande **OpenSSL : openssl dgst -sha256-out HashAuthData AuthData**
- * Vérifiez que **HashAuthData** a été créé correctement → (0.25 pt).
- * Vérifiez le contenu de **HashAuthData** → (0,25 pt).
- * Vous allez maintenant procéder à la signature de **HashAuthData grâce à AlicePrivateKey** en nommant la signature **AliceSignature.** Pour ce faire, veuillez utiliser la commande **OpenSSL openssl rsautl et les options : -sign, -in, -inkey, -out** → (2 pt).
- Pour plus d'informations sur l'option : **-sign**, veuillez utiliser la commande : **openssl rsautl-help.**
- Vérifiez que **AliceSignature** a été créée correctement.
- Copiez **AliceSignature et AuthData dans le dossier de Bob.**
- Retournez dans le dossier parent **LAB1 et accédez au dossier de Bob.**
- Vérifiez que **AliceSignature et AuthData** ont été correctement copiées dans le dossier de **Bob**.
- Vous allez maintenant procéder à la vérification **d'AliceSignature grâce à AlicePublicKey**. Pour ce faire, vous devez de

- Récupérer les **HashAuthData** en entrant la commande **OpenSSL** : **openssl rsautl -verify-in AliceSignature -pubin -inkey AlicePublicKey -out HashAuthData**
- Vérifiez que **HashAuthData** a été récupéré correctement.
- Calculez un nouveau h**achage HashBob sur AuthData grâce à la fonction de hachage SHA256, en** entrant la commande **OpenSSL : openssl dgst -sha256 -out HashBob AuthData**

- Vérifiez que **HashBob** a été créé correctement.
- Comparez **HashAuthData avec HashBob en entrant la commande UNIX : diff HashBobHashAuthData**

- Vous remarquerez que la **commande diff** ne donne aucun résultat (null), car les deux hachages sont identiques.hashs sont égaux. Par conséquent, Bob confirmera l'authenticité d'Alice, la non-répudiation d'**Alice et l'intégrité de AuthData.**
- Vous pouvez essayer de changer quelques caractères d**a**ns **HashBob ou HashAuthData** et exécuter de nouveau la commande **diff** pour voir la différence.